Esta clase va a ser

- grabada

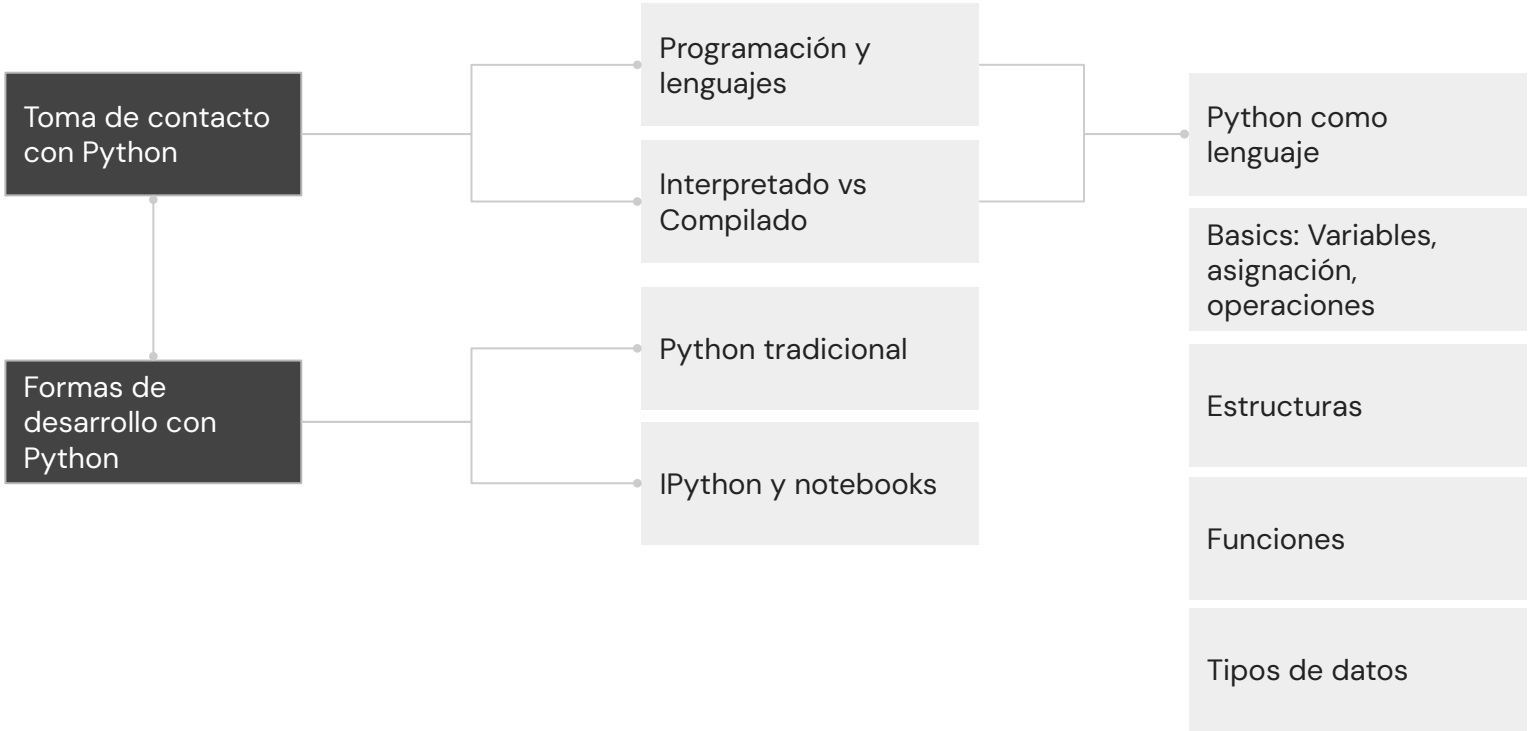
Clase 02. DATA SCIENCE

Hola Python!

Objetivos de la clase

- **Realizar** una primera aproximación al lenguaje de programación Python.
- **Conocer** las distintas formas de desarrollo con Python.

MAPA DE CONCEPTOS



Definición de programa

Desde el principio: programación y Python

¿Qué es la programación?

- ✔ La programación es una forma de ejecutar un **algoritmo**.
- ✔ Un algoritmo es una **secuencia de pasos** que lleva a un **resultado**.
- ✔ Una **receta** es un algoritmo.
- ✔ Si se sigue el algoritmo, se llega al **resultado**.

Programa y computadora

- ✔ La computadora nació para **resolver cálculos**.
- ✔ La programación es un **complemento** para la computadora.
- ✔ Es una forma de que la computadora **entienda el funcionamiento de un algoritmo** y lo ejecute.

- ✔ La computadora entiende **ceros y unos (lenguaje binario)**, nosotros no.
- ✔ Por lo tanto, un programa **traduce un lenguaje humano a lenguaje binario**.

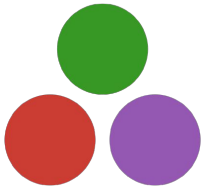
Programación y lenguajes

- ✔ No existe **un solo lenguaje** que solucione **todos** los problemas
- ✔ Cada lenguaje resuelve **un conjunto de problemas posibles:** Empresariales, Web, Ciencia, Salud, etc.

- ✔ Para Data Science, existen algunos lenguajes que funcionan muy bien: **Python, R, Julia y Scala** son algunos de ellos.

Python

R



Julia

Scala

CODERHOUSE

Lenguaje interpretado vs. compilado

¿Interpretado o Compilado?

Python es un lenguaje interpretado, esto quiere decir que:

- ✔ Usa un programa intérprete que traduce en tiempo casi real nuestras órdenes a binario.
- ✔ La traducción se hace línea por línea.
- ✔ Podemos probar código “de a pedacitos”.
- ✔ El lenguaje compilado se traduce todo junto al final.
- ✔ No es simplemente una mejora, es una forma de trabajar muy útil para Data Science.

PARA RECORDAR

Python como lenguaje

Python es el **lenguaje Open Source más solicitado** en las búsquedas laborales relacionadas con **Data Science** y se ubica entre el **segundo y tercer puesto en 2021** de acuerdo a varios rankings de lenguajes de desarrollo general (no sólo Data Science).

Python, Open Source: componentes

1

Intérprete

programa intérprete, traductor a binario.

2

IDE

entorno de desarrollo, lugar donde escribiremos código.

3

Paquetes

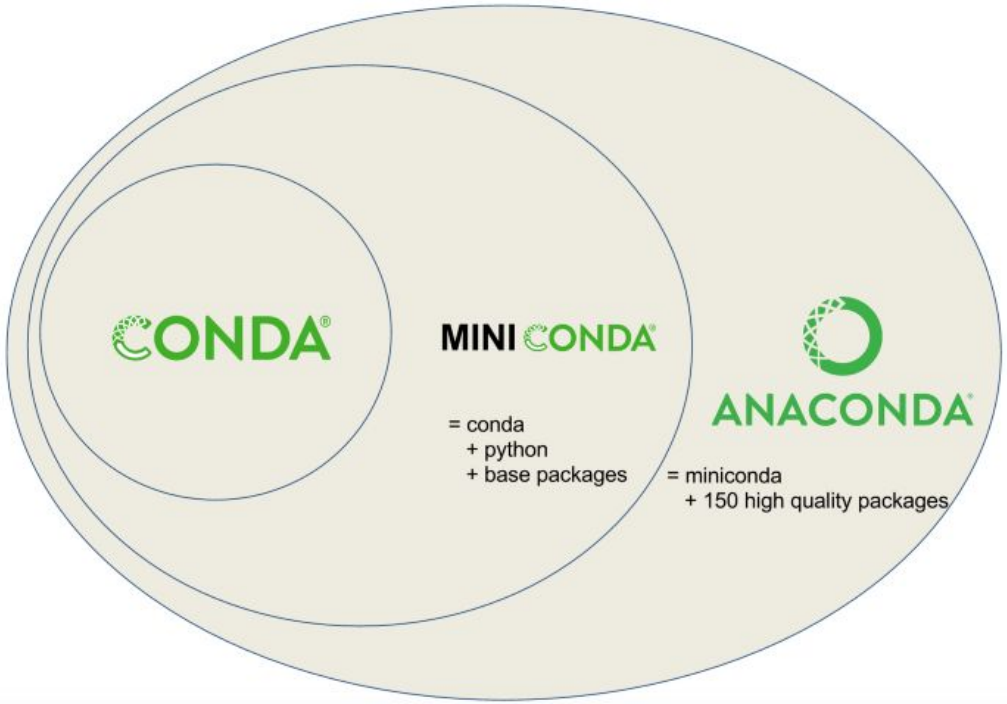
conjuntos de funciones pre-armadas para problemas habituales.

Instalación de Python vía miniconda

Python tradicional

Anaconda y Miniconda

Tradicionalmente, Python puede desarrollarse en Anaconda, o en su versión simplificada, Miniconda.
A su vez, puede utilizarse de varias formas más:



CODERHOUSE

Formas

La forma más básica es escribiendo python en la terminal, lo que abre un entorno de trabajo dentro de la misma terminal.

📢 No es la forma más cómoda, ni la más utilizada.

Otra forma más útil es usando Python interactivo (IPython). Puede accederse escribiendo ipython en la terminal.

📢 No aporta muchas mejoras si se usa de esa forma.

¡No siempre es la mejor forma!

Jupyter Notebooks

IPython y notebooks

Las notebooks siguen siendo IPython, pero con **vitaminas** 🚀

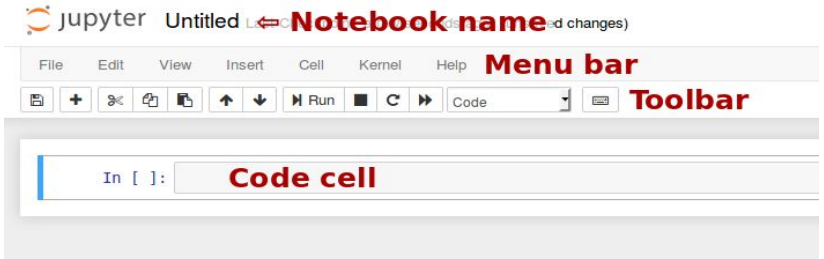
- ✔ Escribimos código en el navegador que resulta ser el IDE.
- ✔ El código pasa por el mismo intérprete que es el que usa la terminal, pero todo se trabaja en el navegador.
- ✔ El código se escribe en cajas de texto que pueden ejecutarse de a una o todas juntas.
- ✔ El conjunto total de cajas de texto es una notebook.

👉 Esta configuración es de las más utilizadas para Data Science.

IPython y notebooks

Podemos encontrar 4 partes principales:

1. Nombre del notebook (termina con extensión .ipynb)
2. Barra de menú: Permite ejecutar código y opciones genéricas
3. Toolbar: Permite ejecutar celdas de código, guardar, añadir, borrar, cortar o pegarlas
4. Celdas de Código: Pueden ser Markdown (texto) o Código Python



¿Cómo usar Google Collab?

Google Colab

Permite trabajar en un entorno no local y la creación de Notebooks 🚀

- ✔ Es un producto de Google Research. Está especialmente adecuado para tareas de aprendizaje automático, análisis de datos y educación.
- ✔ Jupyter es el proyecto de código abierto en el que se basa Colab.
- ✔ Nos permite compartir notebooks sin la necesidad de descargar ningún software extra.
- ✔ El código se ejecuta en una máquina virtual dedicada a tu cuenta y pueden eliminarse luego de cierto tiempo.

Ejemplo en vivo

¿Cómo podemos usar Google Colab como un entorno para programar lenguaje de Python? ¡Vamos a verlo!

IDE s

IDE’s

Son **aplicaciones de software** que permiten a programadores desarrollar código en diferentes lenguajes.
Consta, usualmente, de:

- ✔ Editor de código.
- ✔ Depuradores (Debuggers) que permiten encontrar errores en el código
- ✔ Herramientas automáticas

IDE’s para el desarrollo de Python

Permite trabajar en un entorno no local y la creación de Notebooks 🚀
Las herramientas que mostramos anteriormente no son las únicas en donde compilar código de Python...

CODERHOUSE

¡Atención!

Recuerda instalar Python con Anaconda para la próxima clase.

Ver tutoriales en carpeta de clase

CODERHOUSE

Break

¡10 minutos y volvemos!

Nociones básicas: Variable, asignación, expresiones

Variable

Variables

Las variables se utilizan para almacenar información para ser referenciada y manipulada en un programa de computadora. Proporcionan una forma de etiquetar los datos con un nombre descriptivo, para que los programas puedan ser entendidos con mayor claridad.

Es útil pensar en las variables como contenedores de información.
Su único propósito es etiquetar y almacenar datos en la memoria.

Variables

Los tipos de datos estándar o integrados de Python:

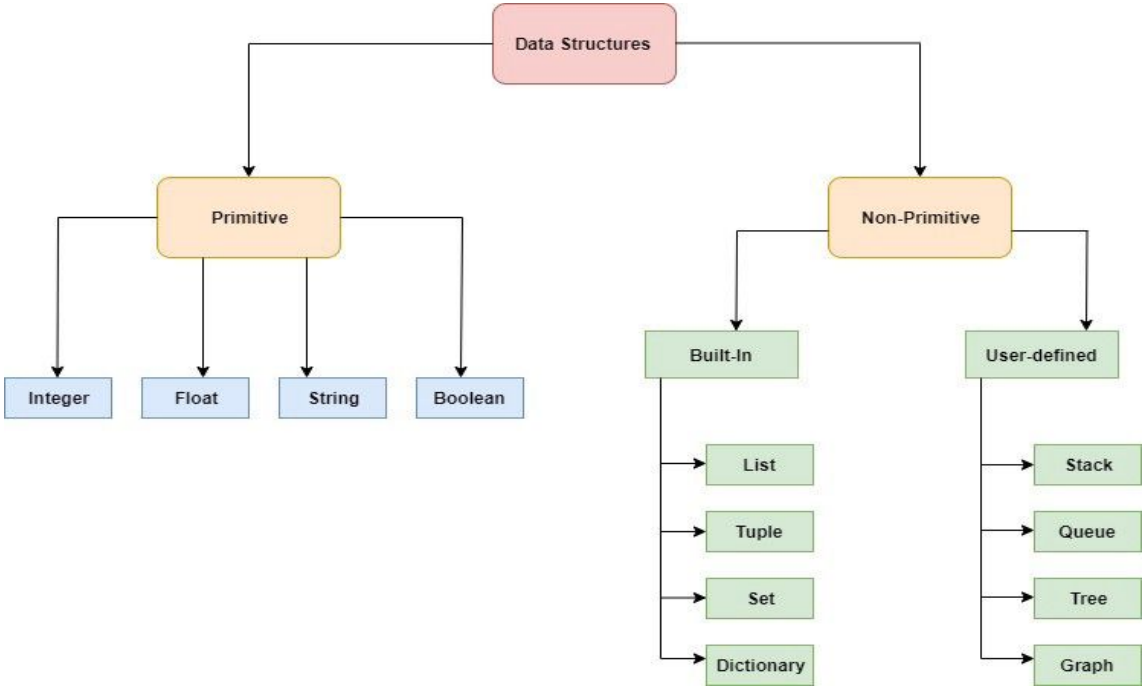
a) Numérico

b) Tipo de secuencia

c) Booleano

d) Conjuntos

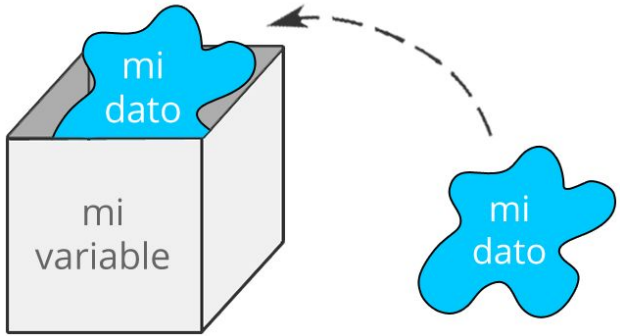
e) Diccionario



CODERHOUSE

Asignación

Asignación

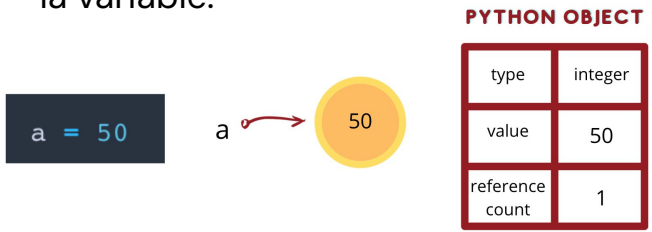


Nombrar variables es una tarea compleja.
Cuando nombre variables, piense detenidamente en los nombres (Comprensible).
La asignación de lleva a cabo por medio del símbolo =
El nombre de la variable va a la izquierda y el valor que desea almacenar en la variable va a la derecha.

Asignación

Reglas para asignación de variables en Python

- ✔ El nombre de una variable debe comenzar con una letra o el carácter de subrayado.
- ✔ Un nombre de variable no puede comenzar con un número.
- ✔ Un nombre de variable solo puede contener caracteres alfanuméricos y guiones bajos (A-z, 0-9 y _).
- ✔ Los nombres de las variables distinguen entre mayúsculas y minúsculas (nombre, Nombre y NOMBRE son tres variables diferentes).
- ✔ Las palabras reservadas (palabras clave) no se pueden usar para nombrar la variable.



CODERHOUSE

Objetos y punteros

Objetos y punteros

Python es un lenguaje orientado a objetos
Es así que en Python todo es un objeto, o sea, cuenta con:

- ✔ Datos
- ✔ Metadatos, atributos o propiedades (un punto y una palabra sin paréntesis): **X.atributo** 👉 Un atributo caracteriza al dato
- ✔ Funcionalidad o métodos (un punto y una palabra con paréntesis): **x.método**() 👉 Un método es algo que el dato puede hacer, por lo tanto al ejecutarlo le estamos pidiendo al dato que ejecute una acción

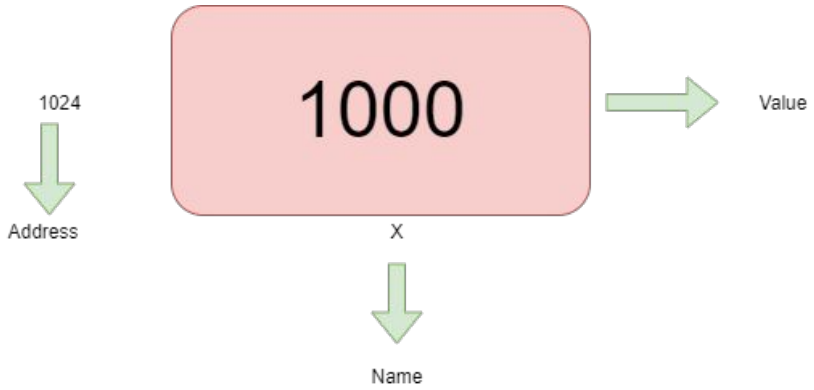
```
x=1.0
x.is_integer() # ¿es x un entero? se lo preguntamos con el método is_integer()

x=1.4
x.is_integer() # ¿y ahora? se lo preguntamos de vuelta

print(x.real,x.imag)  # miramos los atributos de x, en este caso su parte real
                      # y su parte imaginaria
```

Objetos y punteros

Las variables en Python **no contienen los datos, sino que apuntan a los datos.**
Esta es la forma de trabajo de los **punteros**, lo que hace que el lenguaje sea más eficiente.



CODERHOUSE

Para pensar

¿Cuáles son las salidas de los siguientes bloques de código?

```
In [ ]: x=1
        type(x)

In [ ]: print(x)
        x= 'Hola Mundo'
        print(x)

In [ ]: x=[1,2,3]
        y= x
        print(y,x)

In [ ]: x.append(4)
        print(y)

In [ ]: x=5
        print(x,y)
```

Contesta mediante el chat de Zoom

CODERHOUSE

In [ ]:
```
x = 1
type(x)
```

In [ ]:
```
print(x)
x="hola"
print(x)
```

In [ ]:
```
x = [1, 2, 3]
y = x
print(y,x)
```

In [ ]:
```
x.append(4)
print(y)
```

In [ ]:
```
x=5
print(x,y)
```

¿No notaste algo raro en el ejercicio anterior...?

CODERHOUSE

Objetos y punteros

Cuando operamos sobre una variable (método) **operamos sobre el objeto al que apunta.**

Cuando realizamos una asignación (=) **conectamos (apuntamos) la variable al objeto**. Aquí no cambiamos el objeto.

Objetos y punteros

```
x = [1, 2, 3]      # x es una lista
y = x              # el objeto al que apunta x ([1, 2, 3]) ahora es también
                   # apuntado por y
print(y is x)      # x e y son el mismo objeto (True)
print(x,y)         # [1, 2, 3] [1, 2, 3]
x.append(4)        # aquí operó sobre el objeto [1, 2, 3] apuntado por x.
                   # Los métodos se identifican luego de un punto (x.método())
print(y)           # como x e y apuntan al mismo objeto, y refleja los cambios
x = "hola"         # al realizar asignación, ahora x apunta al objeto texto
                   # (string) "hola"
print(x is y)      # x e y ahora no apuntan al mismo objeto (False)
print(x,y)         # x e y apuntan a dos objetos diferentes (“hola” [1, 2, 3, 4])
```

Objetos y punteros

La diferencia es **muy sutil** y en general no afecta el trabajo de Data Science. No obstante, **no todos los lenguajes se comportan así.**

⚠️ Hay que tener en cuenta esto para no cometer errores.

- ✔ Un método comienza por un **punto después de la variable.**
- ✔ El método **modifica el objeto apuntado** por la variable.
- ✔ La variable **no es, ni contiene al objeto.**
- ✔ La asignación **“conecta” a la variable con el objeto apuntado.**

Expresiones

Expresiones

Una expresión es una combinación de operadores y operandos que se interpreta para producir algún otro valor.
En cualquier lenguaje de programación, una expresión se evalúa según la precedencia de sus operadores.

	expression	value	type
literal	500	500	integer
	3.14	3.14	float
+	200 + 300	500	integer
	10.0 + 5.0	15.0	float

CODERHOUSE

Expresiones

Expresiones constantes: son las expresiones que solo tienen valores constantes.

```
x = 15 + 1.3
print(x)
```

```
16.3
```

Expresiones

Expresiones aritméticas: una expresión aritmética es una combinación de valores numéricos, operadores y, a veces, paréntesis.

```
x = 40
y = 12
 add = x + y
sub = x - y
pro = x * y
div = x / y
print(add);print(sub);print(pro);print(div)
```

```
52
28
480
3.3333333333333335
```

CODERHOUSE

Expresiones

Expresiones integrales: este es el tipo de expresiones que producen solo resultados enteros después de todos los cálculos.

```
a = 13
b = 12.0
 c = a + int(b)
print(c)
```

```
25
```

Expresiones

Expresiones flotantes: este es el tipo de expresiones que producen números de punto flotante como resultado de todos los cálculos

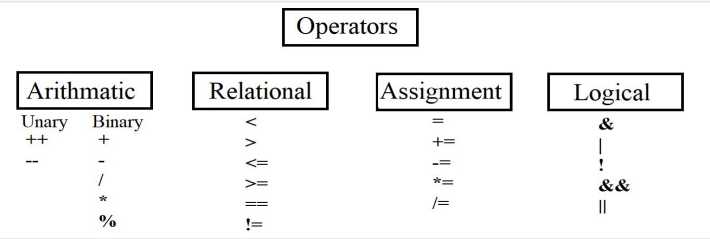
```
a = 13
b = 5
 c = a / b
print(c)
```

```
2.6
```

CODERHOUSE

Operadores

Operadores



- Los operadores permiten **trabajar sobre las variables**, a la manera de las operaciones matemáticas.
- Cada operador da un **resultado** como salida.
- Identificamos 4 tipo de operadores:
  - Operadores aritméticos
  - Operadores de asignaciones
  - Operadores de identidad y pertinencia

Operadores aritméticos

Los **operadores aritméticos** son directamente operaciones matemáticas estándar.

Aritméticos	
a + b	Suma
a - b	Resta
a * b	Multiplicación
a / b	División
a // b	División entera (resultado sin decimal)
a % b	Módulo (resto de la división entera)
a ** b	Exponenciación
-a	Negativo

Python Operator Precedence

Precedence	Operator Sign	Operator Name
Highest	**	Exponentiation
	+x, -x, ~x	Unary positive, unary negative, bitwise negation
	*, /, //, %	Multiplication, division, floor, division, modulus
	+, -	Addition, subtraction
	<<, >>	Left-shift, right-shift
	&	Bitwise AND
	^	Bitwise XOR
	|	Bitwise OR
	==, !=, <, <=, >, >=, is, is not	Comparison, identity
	not	Boolean NOT
	and	Boolean AND
Lowest	or	Boolean OR

Operadores de asignaciones

Los asignadores simplifican operadores aritméticos comunes.

Asignaciones	
a += b	a = a + b
a -= b	a = a - b
a *= b	a = a * b
a /= b	a = a / b
a //= b	a = a // b
a %= b	a = a % b
a **= b	a = a ** b

Comparadores	
a == b	a igual a b
a != b	a distinto de b
a < b	a menor a b
a > b	a mayor a b
a <= b	a menor o igual que b
a >= b	a mayor o igual que b

Operadores de comparación

Los comparadores dan resultados lógicos **(si/no, true/false)**

CODERHOUSE

Operadores de identidad y pertenencia

- ✔ Los operadores de identidad y pertenencia verifican relaciones entre objetos.
- ✔ Dentro de esta categoría, los operadores “in”, como casos particulares, buscan objetos dentro de listas. ¡Son muy útiles!

Identidad y pertenencia	
a is b	a es el mismo objeto que b
a is not b	a no es el mismo objeto que b
a in b	a está contenido en b
a not in b	a no está contenido en b

CODERHOUSE

Uso de filtros booleanos

Los operadores nos permiten crear filtros booleanos que ayudan a obtener filtros rápidos para información de interés

```
import pandas as pd
import numpy as np
df= pd.DataFrame(data=np.random.randint(64,
size=(8,8)),columns=['Ja','Mu','Ct','Dn','Eo','Tp','Yn','Om'])
print(df)
```

	Ja	Mu	Ct	Dn	Eo	Tp	Yn	Om
0	26	41	62	50	41	52	49	35
1	4	13	37	4	18	7	30	43
2	57	19	41	55	53	15	14	57
3	47	6	53	47	58	15	39	49
4	61	32	53	15	48	56	42	20
5	60	56	40	55	7	32	51	13
6	45	9	29	21	48	43	61	30
7	44	5	27	59	6	47	7	46

```
index_bool=df['Eo']>10
index_bool
```

```
df['Eo'][index_bool]
0    41
1    18
2    53
3    58
4    48
6    48
Name: Eo, dtype: int64
```

CODERHOUSE

Ejemplo en vivo

Examinemos un poco lo que se conoce como estructuras de control

5 minutos

Para pensar

¿Qué diferencia hay entre usar and/or? ¿Que significa el operador %?

```
In [1]: # Extructura de control (intro)
        x= 5
        print(x>5)

        False

In [2]: print(x>1 and x <6)

        True

In [3]: print(x>1 or x <6)

        True

In [5]: 3 < x <= 5

Out[5]: True

In [6]: x % 3 == 0 or x % 5 == 0

Out[6]: True
```

Contesta mediante el chat de Zoom

CODERHOUSE

¿Preguntas?

CLASE N°3

Glosario

Programación: formas de ejecutar un algoritmo (recetas)

Lenguajes: herramientas computacionales que permiten resolver problemas con estructuras de código. En Data Science existen varios comunes: Python , R , Java, Julia, C, C++

Lenguaje interpretado: cualquier lenguaje de programación que se ejecute línea a línea y que convierta las órdenes a formato binario (e.g Python , R)

IDE: aplicaciones donde escribimos el código de un lenguaje particular (e.g Spyder, Kite, Visual Studio, Atom)

Variable: Cualquier estructura que permita almacenar información para su manipulación

Asignación: Proceso mediante el cual se le asigna un valor particular a una variable

Punteros: herramientas que nos permiten conectar a las variables con sus valores respectivos

Expresiones: combinaciones de operadores y operandos que dan como resultado un valor particular

Operadores: son los que permiten trabajar sobre las variables, pueden ser de 4 tipos (aritméticos, relacionales, de asignación y lógicos)

Resumen
de la clase hoy

- ✓ Definición de Programa
- ✓ Lenguaje Interpretado vs compilado
- ✓ Python como Lenguaje
- ✓ Nociones básicas: variable, asignación y expresiones
- ✓ Objetos y punteros

Opina y valora esta clase

Muchas gracias.

#DemocratizandoLaEducación